CHAPTER 3

CREATING TABLES

LEARNING OBJECTIVES

Objectives

- Create and run SQL commands
- Create tables
- Identify and use data types to define columns in tables
- Understand and use nulls
- Add rows to tables
- View table data
- Correct errors in a table
- Save SQL commands to a file
- Describe a table's layout using SQL

INTRODUCTION

You already might be an experienced user of a database management system (DBMS). You might find a DBMS at your school's library, at a site on the Internet, or in any other place where you retrieve data using a computer. In this chapter, you will begin your study of **Structured Query Language (SQL)**, which is one of the most popular and widely used languages for retrieving and manipulating database data.

In the mid-1970s, SQL was developed as the data manipulation language for IBM's prototype relational model DBMS, System R, under the name SEQUEL at IBM's San Jose research facilities. In 1980, the language was renamed SQL (but still pronounced "sequel" although the equally popular pronunciation of "S-Q-L" ["ess-cue-ell"] is used in this text) to avoid confusion with an unrelated hardware product named SEQUEL. Most DBMSs use a version of SQL as their data manipulation language.

In this chapter, you will learn the basics of working in SQL. You will learn how to create tables and assign data types to columns. You also will learn about a special type of value, called a null value, and learn how to manage these values in tables. You will learn how to insert data into your tables after you create them. Finally, you will learn how to describe a table's layout using SQL.

CREATING AND RUNNING SQL COMMANDS

You accomplish tasks in SQL by creating and running commands. In order to do so, you need to use a DBMS that supports SQL. This text uses Oracle as the DBMS in which to create and run the commands. The text also indicates differences you will find if you are using Microsoft Access or Microsoft SQL Server 2005. (If you are using MySQL, contact Cengage Learning for the latest edition of *A Guide to MySQL*, by Pratt and Last.)

Although the version of Oracle used in this text is the Oracle Database 10*g* Express Edition, the commands used in this text will work the same in any other version of Oracle. You use the Oracle Database Express Edition by downloading it from the Oracle Web site, installing it, and then starting it using the Microsoft Internet Explorer Web browser.

Starting the Oracle Database Express Edition

After installing the Oracle Database Express Edition, you start it by clicking the Start button, pointing to All Programs, clicking Oracle Database 10*g* Express Edition, and then clicking Go To Database Home Page. Internet Explorer will start and load the home page, which requests your username and password. (Ask your instructor which username and password to use, or use the one you specified when you installed the software. If a different Web browser starts, ask your instructor for help. Other Web browsers might not fully support the examples used in this text.) After entering this information, click the Login button. Figure 3-1 shows the Oracle Database Express Edition home page. You click the icons on the home page to access the various tools. In this text, you will use the SQL tool. The other tools let you administer a database, work with database objects, and run different database utilities. (These features are beyond the scope of this book.)

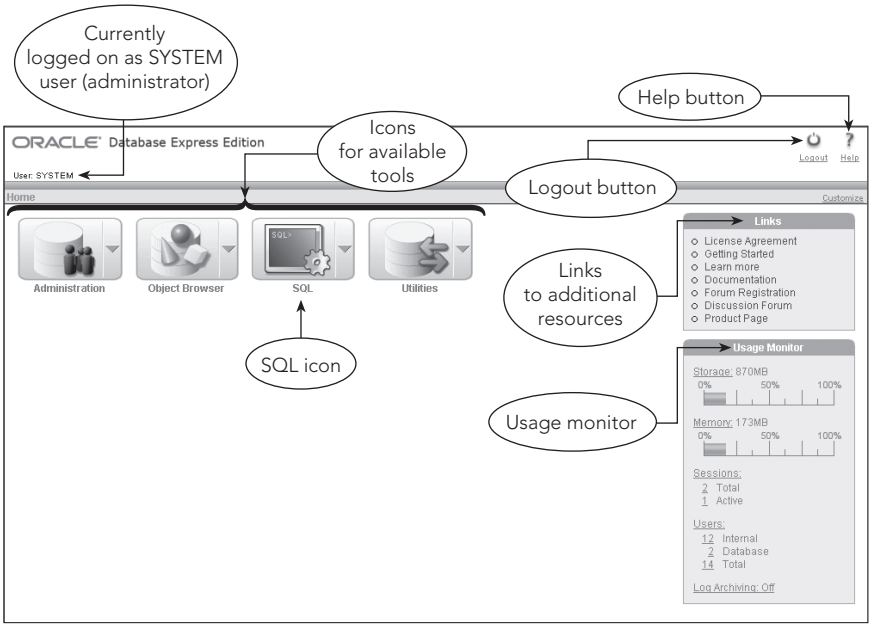

FIGURE 3-1 Oracle Database Express Edition home page

NOTE

The figure shows that the SYSTEM (administrator) user is currently logged on. Your instructor might assign you another username to use as your login, in which case, this name will appear on your screen.

There are two ways to use the tools in the Oracle Database Express Edition. You can click the arrow for the icon to display a menu and then select an option from the menu. Figure 3-2 shows the result of clicking the arrow for the SQL icon and then pointing to the SQL Scripts option on the SQL menu. A submenu of commands for working with SQL scripts appears. To create a script using this approach, for example, you would click the arrow, point to SQL Scripts, and then click Create. (You will learn more about scripts later in this chapter.)

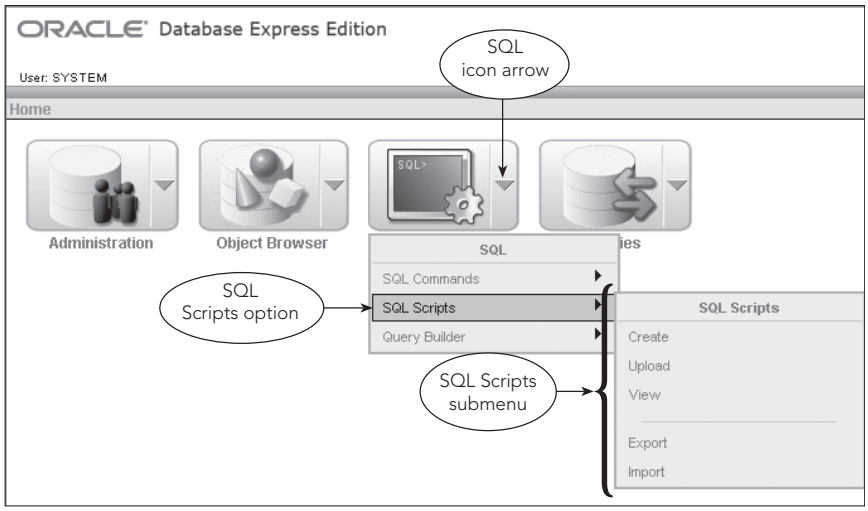

FIGURE 3-2 SQL Scripts submenu

You also can click the icon to display the options as icons instead of as submenus. For example, when you click the SQL icon on the home page, you will see the SQL page shown in Figure 3-3. Clicking an arrow on an icon displays a submenu. The figure shows the results of clicking the arrow for the SQL Scripts icon, which displays the SQL Scripts submenu. To create a script using this approach, click the SQL icon on the home page, click the arrow for the SQL Scripts icon, and then click Create. The approach you choose is a matter of personal preference.

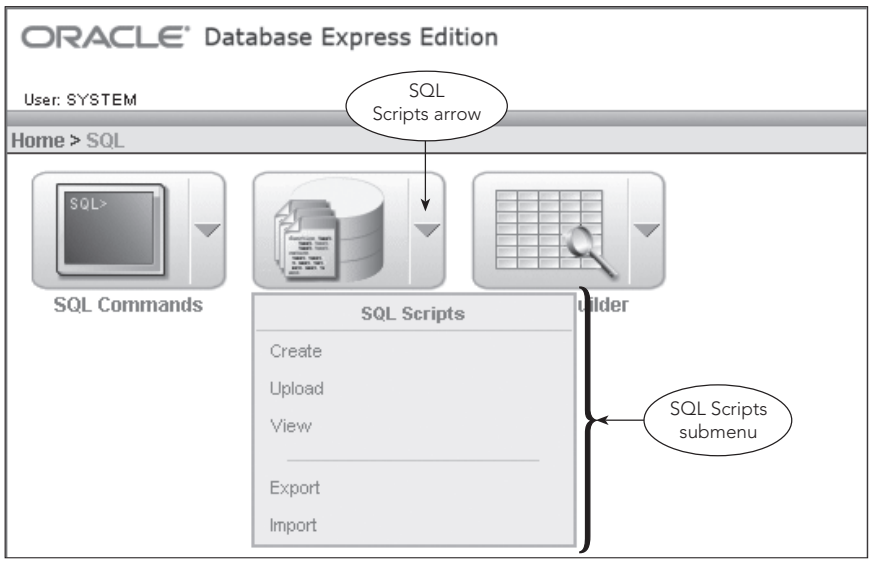

FIGURE 3-3 SQL Scripts submenu on the SQL page

Entering Commands

You enter commands on the SQL Commands page. To access the SQL Commands page, click the arrow for the SQL icon, and then point to SQL Commands as shown in Figure 3-4.

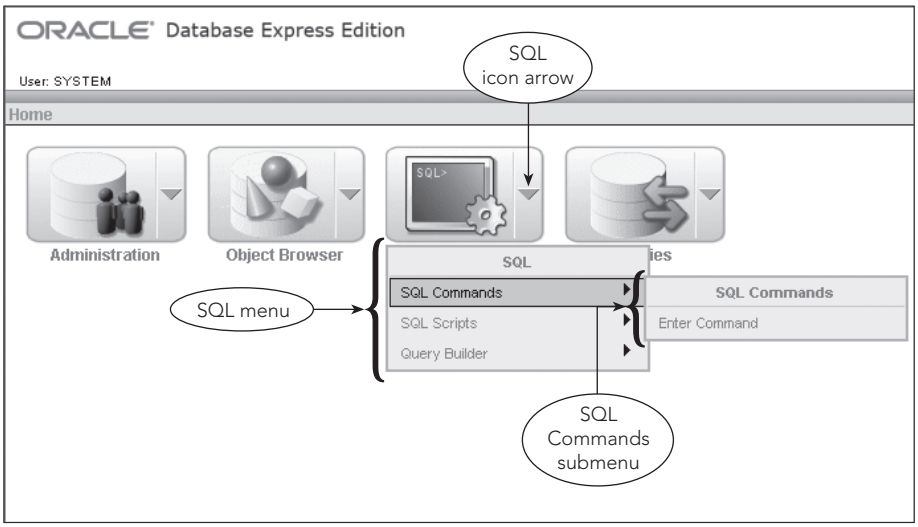

FIGURE 3-4 Starting a new SQL command

After clicking the Enter Command option on the SQL Commands submenu, you will see the SQL Commands page shown in Figure 3-5. You enter the command in the upper portion of this page, called the SQL editor pane, and then click the Run button to execute the command and display its results in the lower portion of the page, called the Results pane.

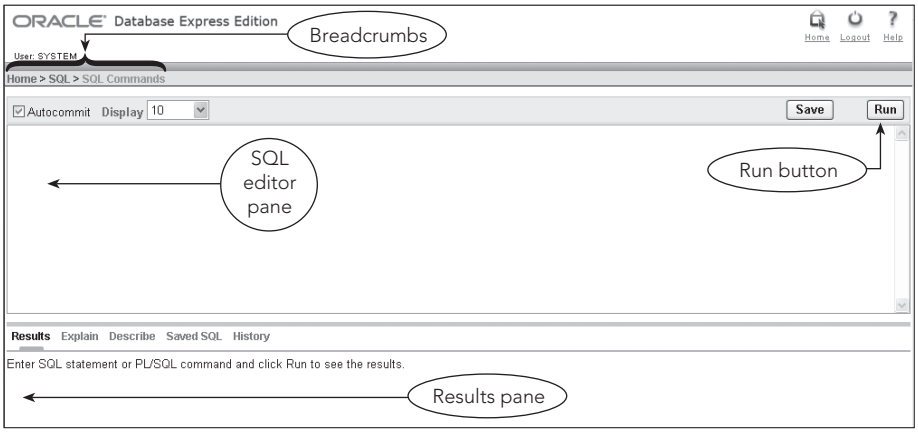

FIGURE 3-5 SQL Commands page

After clicking the Run button, the results will appear in the Results pane. Notice the Home> SQL> SQL Commands reference at the top of the SQL editor pane. This reference is called a **breadcrumb**. You can click the pages in the breadcrumb to move back one or more pages. For example, to return to the home page, click Home in the breadcrumb.

CREATING A TABLE

Before you begin adding data to a table, you must describe the layout of the table to the DBMS.

EXAMPLE 1

Describe the layout of the REP table to the DBMS.

You use the **CREATE TABLE** command to describe the layout of a table. The word TABLE is followed by the name of the table to be created and then by the names and data types of the columns that the table contains. The **data type** indicates the type of data that the column can contain (for example, characters, numbers, or dates) as well as the maximum number of characters or digits that the column can store.

The restrictions placed on table and column names are as follows:

1. The names cannot exceed 30 characters.
2. The names must start with a letter.
3. The names can contain letters, numbers, and underscores (_).
4. The names cannot contain spaces.

The SQL command that creates the REP table is shown in Figure 3-6.

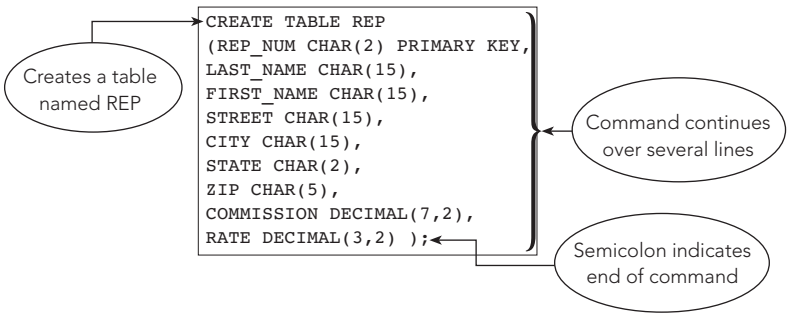

FIGURE 3-6 CREATE TABLE command for the REP table

This CREATE TABLE command, which uses the data definition features of SQL, describes a table named REP. The table contains nine columns: REP_NUM, LAST_NAME, FIRST_NAME, STREET, CITY, STATE, ZIP, COMMISSION, and RATE. The REP_NUM column can store two characters and is the table's primary key. The LAST_NAME column can store 15 characters, and the STATE column can store two characters. The COMMISSION column can store only numbers, and those numbers are limited to seven digits, including two decimal places. Similarly, the RATE column can store three numbers, including two

decimal places. You can think of the SQL command shown in Figure 3-6 as creating an empty table with column headings for each column name.

In SQL, commands are free format; that is, no rule says that a particular word must begin in a particular position on the line. For example, you could have written the CREATE TABLE command shown in Figure 3-6 as follows:


```
CREATE TABLE REP (REP_NUM CHAR(2) PRIMARY KEY, LAST_NAME
CHAR(15), FIRST_NAME CHAR(15), STREET CHAR(15), CITY
CHAR(15), STATE CHAR(2), ZIP CHAR(5), COMMISSION
DECIMAL(7,2), RATE DECIMAL(3,2) );
```

The manner in which the CREATE TABLE command shown in Figure 3-6 was written makes the command more readable. This text will strive for such readability when writing SQL commands.

NOTE

SQL is not case sensitive; you can type commands using uppercase or lowercase letters. There is one exception to this rule, however. When you are inserting character values into a table, you must use the correct case.

To create the REP table in Oracle, click in the SQL editor pane, type the CREATE TABLE command shown in Figure 3-7, and then click the Run button on the right side of the SQL editor pane to execute the command and create the table. Figure 3-7 also shows the message that appears in the Results pane after running the command, which indicates that the table was created.

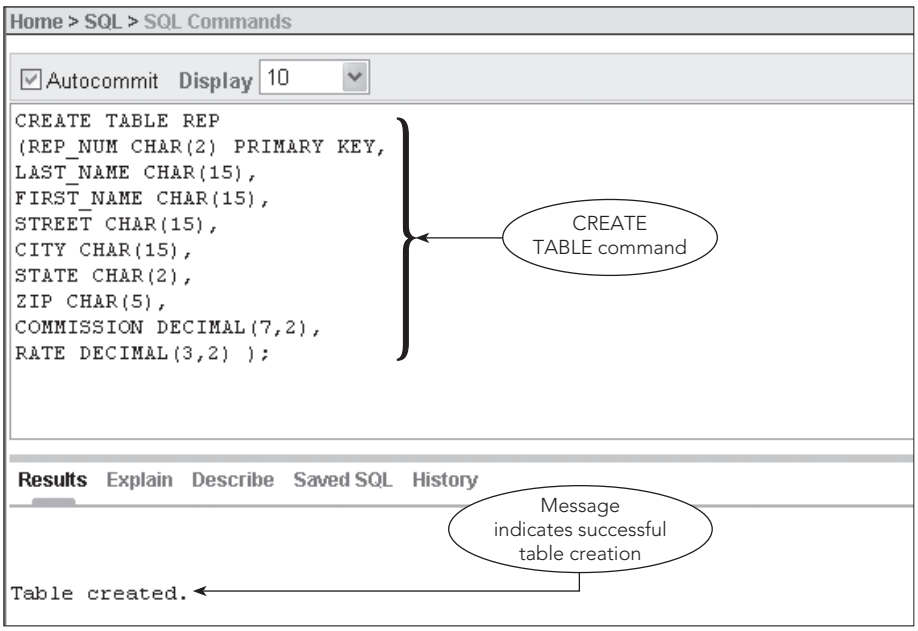

FIGURE 3-7 Running the CREATE TABLE command for the REP table

ACCESS USER NOTE

Microsoft Office Access is a DBMS that lets you work in a graphical user interface, but you can also use it to run SQL commands. To run SQL commands in Access, you must first create a new query. In Access 2007, open the database, click the Create tab on the Ribbon, click the Query Design button in the Other group, close the Show Table dialog box, and then click the SQL View button in the Results group. (In Access 2003, open the database, and then create a new query in Design view. Close the Show Table dialog box, and then click the SQL View button on the Query Design toolbar.) In the Query window, type the SQL command, and then click the Run button on the Query Design toolbar (Access 2003) or the Run button in the Results group on the Design tab (Access 2007) to execute the command. Figure 3-8 shows the command to create the REP table using Access 2007. When you click the Run button, Access will create the table shown in the CREATE TABLE command. (Unlike Oracle, Access doesn't display a message indicating the result was successful.)

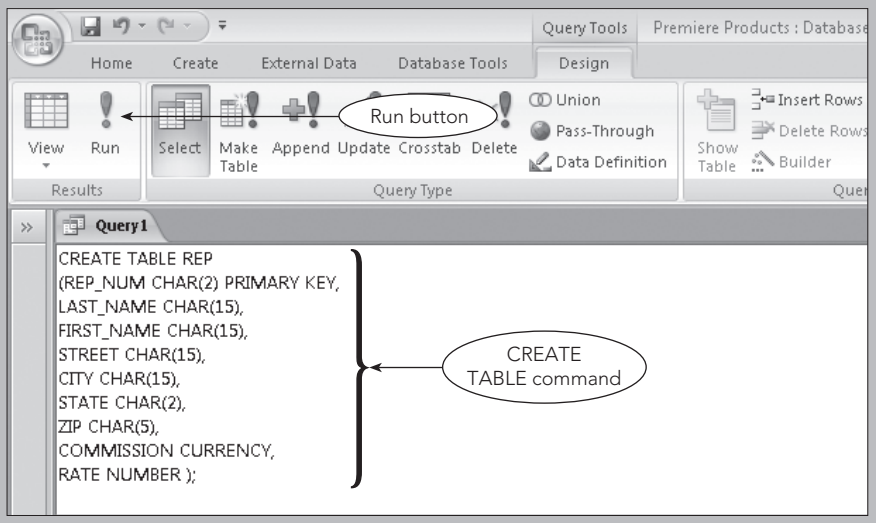

FIGURE 3-8 Using Access SQL view to create a table

Unlike Oracle, Access does not support the DECIMAL data type. To create numbers with decimals, you must use either the CURRENCY or NUMBER data type. Use the CURRENCY data type for fields that will contain currency values; use the NUMBER data type for all other numeric fields.

In Access, it is common to create a table using Table Design view and then to add records to the table using Datasheet view. You still can run SQL commands when you create tables using Design view and enter data into them using Datasheet view.

SQL SERVER USER NOTE

Microsoft SQL Server 2005 is a DBMS designed for use in client-server applications. You can run Microsoft SQL Server 2005 from your own computer through a set of client database tools called SQL Server Management Studio. Management Studio includes a Query Editor window that you can use to run SQL commands. If you are using Management Studio and connecting to a database on your local computer, accept the default values for Server Type, Server Name, and Authentication, and then click the Connect button in the Connect to Server dialog box. When Management Studio appears, double-click Databases, open the database on which you want to run SQL commands, and click the New Query button on the toolbar. Type the SQL command in the Query Editor window that opens, and then click the Execute button on the toolbar to execute the command. The command shown in Figure 3-9 creates the REP table and displays a message in the Messages pane to indicate that the command completed successfully.

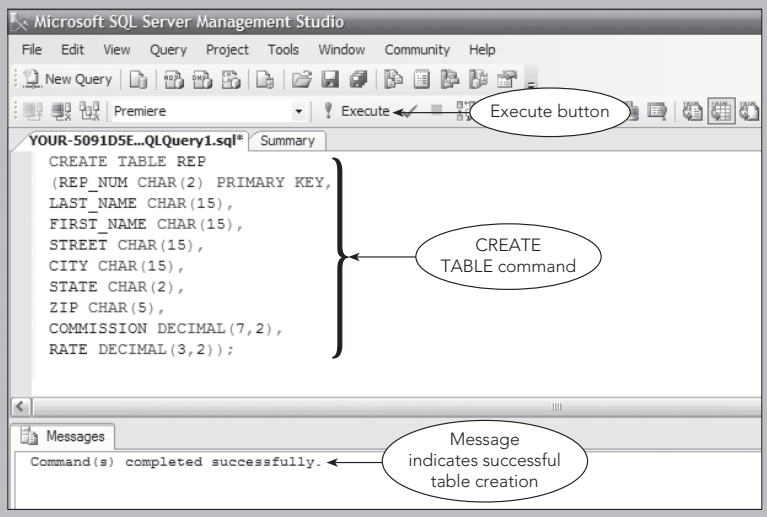

FIGURE 3-9 Using Microsoft SQL Server 2005 to create a table

Correcting Errors in SQL Commands

Suppose that you executed the REP table using the CREATE TABLE command shown in Figure 3-10, which contains several mistakes. Instead of displaying a message that the table was created successfully, Oracle displays an error message about a problem that it encountered. In reviewing the command, you see that CHAR is misspelled on line 4, line 5 is missing a comma, the CITY column was omitted, and line 7 should be deleted. If you run an SQL command and Oracle displays an error, you can use the mouse and the arrow keys on the keyboard to position the insertion point in the correct position so you can correct these errors using the same techniques that you might use in a word processor. For example, you can use the pointer to select the word CHR on line 4 and type CHAR. Then you can use the pointer to move the insertion point to the end of line 5 so you can type the missing comma, and then press Enter to insert the missing information to create the CITY column. You can use the pointer to select the contents of line 7 and then press Delete to remove it. After making these changes, you can click the Run button to execute the command again. If the command contains additional errors, you'll see an error message again. If the command is correct, you'll see the message that the table was created.

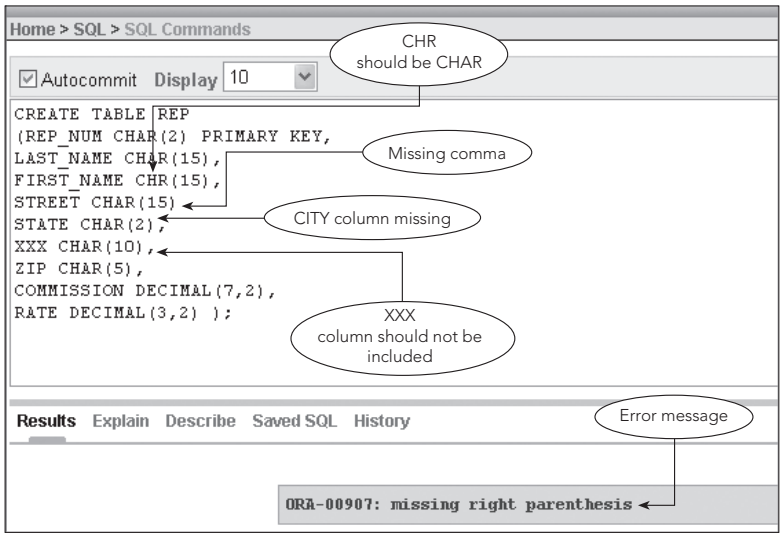

FIGURE 3-10 CREATE TABLE command with errors

Dropping a Table

After creating a table, you might notice that you added a column that you do not need or that you assigned the wrong data type or size to a column. Another way of correcting errors in a table is to delete (drop) the table and start over. For example, suppose you wrote a CREATE TABLE command that contained a column named LST instead of LAST or defined a column as CHAR(5) instead of CHAR(15). Suppose you do not discover the error and you execute the command, creating a table with these problems. In this case, you can delete the entire table using the **DROP TABLE** command and then re-create the table using the correct CREATE TABLE command.

To drop a table, execute the DROP TABLE command, followed by the name of the table you want to delete and a semicolon. To delete the REP table, for example, you would enter the following command and then click the Run button:

```
DROP TABLE REP;
```

Dropping a table also deletes any data that you entered into the table. It is a good idea to check your CREATE TABLE commands carefully before executing them and to correct any problems before adding data. Later in this text, you will learn how to change a table's structure without having to delete the entire table.

Q & A

Question: How can I correct a mistake that I made when I created a table?
Answer: Later in the text, you will see how to alter a table to make any necessary corrections. For now, the easiest way is to drop the table using the DROP TABLE command and then to execute the correct CREATE TABLE command.

USING DATA TYPES

For each column in a table, you must specify the **data type** to use to store the type of data that the column will contain. Figure 3-11 describes some common data types used in databases.

Data type	Description
CHAR(n)	Stores a character string n characters long. You use the CHAR data type for columns that contain letters and special characters and for columns containing numbers that will not be used in any calculations. Because neither sales rep numbers nor customer numbers will be used in any calculations, for example, the REP_NUM and CUSTOMER_NUM columns are both assigned the CHAR data type.
VARCHAR(n)	An alternative to CHAR that stores a character string up to n characters long. Unlike CHAR, only the actual character string is stored. If a character string 20 characters long is stored in a CHAR(30) column, for example, it will occupy 30 characters (20 characters plus 10 blank spaces). If it is stored in a VARCHAR(30) column, it will only occupy 20 spaces. In general, tables that use VARCHAR instead of CHAR occupy less space, but the DBMS does not process them as rapidly during queries and updates. However, both are legitimate choices. This text uses CHAR, but VARCHAR would work equally well.
DATE	Stores date data. The specific format in which dates are stored varies from one SQL implementation to another. In Oracle, dates are enclosed in single quotation marks and have the format DD-MON-YYYY (for example, '15-OCT-2010' is October 15, 2010). In Access, dates are enclosed in number signs and are entered using the format MM/DD/YYYY (for example, #10/15/2010# is October 15, 2010). In SQL Server, use the DATETIME data type to store dates.
DECIMAL(p,q)	Stores a decimal number p digits long with q of these digits being decimal places to the right of the decimal point. For example, the data type DECIMAL(5,2) represents a number with three places to the left and two places to the right of the decimal (for example, 100.00). You can use the contents of DECIMAL columns in calculations. You also can use the NUMBER (p,q) data type in both Oracle and SQL Server to store a decimal number. Access does not support the DECIMAL data type; use the CURRENCY or NUMBER data type instead.
INT	Stores integers, which are numbers without a decimal part. The valid range is -2147483648 to 2147483647. You can use the contents of INT columns in calculations. If you follow the word INT with AUTO_INCREMENT, you create a column for which SQL will automatically generate a new sequence number each time you add a new row. This would be the appropriate choice, for example, when you want the DBMS to generate a value for a primary key.
SMALLINT	Stores integers, but uses less space than the INT data type. The valid range is -32768 to 32767. SMALLINT is a better choice than INT when you are certain that the column will store numbers within the indicated range. You can use the contents of SMALLINT columns in calculations.

FIGURE 3-11 Commonly used data types

USING NULLS

Occasionally, when you enter a new row into a table or modify an existing row, the values for one or more columns are unknown or unavailable. For example, you can add a customer's name and address to a table even though the customer does not have an assigned sales rep or an established credit limit. In other cases, some values might never be known—perhaps there is a customer that does not have a sales rep. In SQL, you handle this situation by using a special value to represent cases in which an actual value is unknown, unavailable, or not applicable. This special value is called a **null data value**, or simply a **null**. When creating a table, you can specify whether to allow nulls in the individual columns.

Q & A

Question: Should a user be allowed to enter null values for the primary key?
Answer: No. The primary key is supposed to uniquely identify a given row, and this would be impossible if nulls were allowed. For example, if you stored two customer records without values in the primary key column, you would have no way to tell them apart.

In SQL, you use the **NOT NULL** clause in a CREATE TABLE command to indicate columns that *cannot* contain null values. The default is to allow nulls; columns for which you do not specify NOT NULL can accept null values.

For example, suppose that the LAST_NAME and FIRST_NAME columns in the REP table cannot accept null values, but all other columns in the REP table can. The following CREATE TABLE command accomplishes this goal:

```
CREATE TABLE REP
(REP_NUM CHAR(2) PRIMARY KEY,
LAST_NAME CHAR(15) NOT NULL,
FIRST_NAME CHAR(15) NOT NULL,
STREET CHAR(15),
CITY CHAR(15),
STATE CHAR(2),
ZIP CHAR(5),
COMMISSION DECIMAL(7,2),
RATE DECIMAL(3,2) );
```

If you created the REP table with this CREATE TABLE command, the DBMS would reject any attempt to store a null value in either the LAST_NAME or FIRST_NAME column. The system would accept an attempt to store a null value in the STREET column, however, because the STREET column can accept null values. Because the primary key column cannot accept null values, you do not need to specify the REP_NUM column as NOT NULL.

ADDING ROWS TO A TABLE

After you have created a table in a database, you can load data into the table by using the INSERT command.

The INSERT Command

The **INSERT** command adds rows to a table. You type INSERT INTO followed by the name of the table into which you are adding data. Then you type the word VALUES followed by the specific values to be inserted in parentheses. When adding rows to character columns, make sure you enclose the values in single quotation marks (for example, 'Kaiser'). You also must enter the values in the appropriate case, because character data is stored exactly as you enter it.

NOTE

You must enclose values in single quotation marks for any column whose type is character (CHAR), even when the data contains numbers. Because the ZIP column in the REP table has a CHAR data type, for example, you must enclose zip codes in single quotation marks, even though they are numbers.

If you need to enter an apostrophe (single quotation mark) into a column, you type two single quotation marks. For example, to enter the name O'Toole in the LAST_NAME column, you would type 'O''Toole' as the value in the INSERT command.

EXAMPLE 2

Add sales rep 20 to the REP table.

The command for this example is shown in Figure 3-12. Note that the character strings ('20','Kaiser','Valerie', and so on) are enclosed in single quotation marks. When you execute the command, the record is added to the REP table.

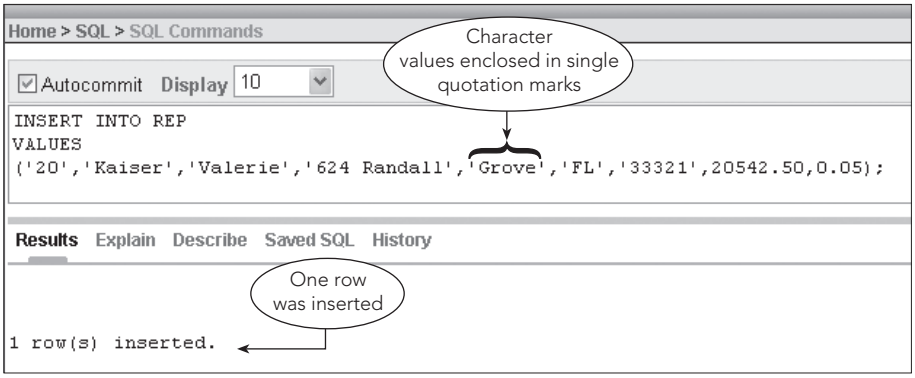

FIGURE 3-12 INSERT command for the first record in the REP table

NOTE

Make sure that you type the values in the same case as those shown in the figures to avoid problems later when retrieving data from the database.

EXAMPLE 3

Add sales reps 35 and 65 to the REP table.

You could enter and execute new INSERT commands to add the new rows to the table. However, an easier and faster way to add these new rows to the table is to use the mouse and the keyboard to modify the previous INSERT command and execute it to add the record for the second sales rep, as shown in Figure 3-13.

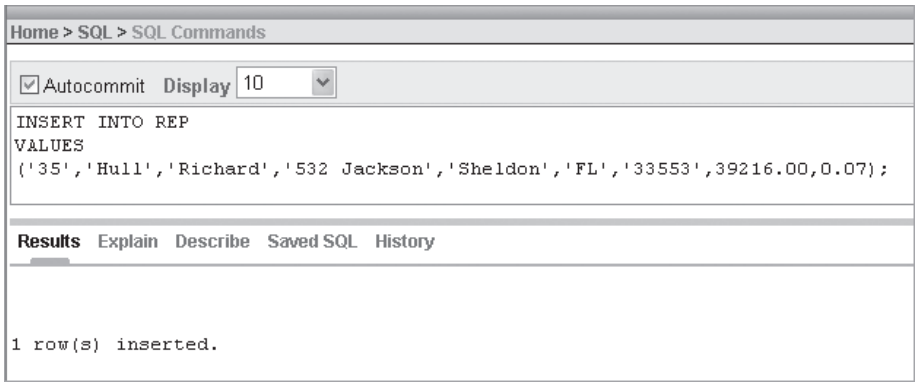

FIGURE 3-13 INSERT command to add the second row to the REP table

You can modify and execute the INSERT command again for the third sales rep, as shown in Figure 3-14, to add the third row to the table.

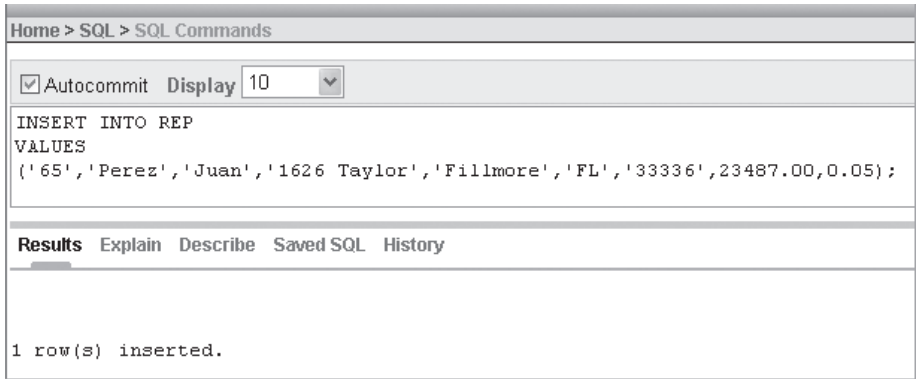

FIGURE 3-14 INSERT command to add the third row to the REP table

Inserting a Row that Contains Nulls

To enter a null value into a table, you use a special form of the INSERT command in which you identify the names of the columns that will accept non-null values, and then list only these non-null values after the VALUES command, as shown in Example 4.

EXAMPLE 4

Add sales rep 85 to the REP table. Her name is Tina Webb. All columns except REP_NUM, LAST_NAME, and FIRST_NAME are null.

In this case, you do not enter a value of *null*; you enter only the non-null values. To do so, you must indicate precisely which values you are entering by listing the corresponding columns as shown in Figure 3-15. The command shown in Figure 3-15 indicates that you are entering data in only the REP_NUM, LAST_NAME, and FIRST_NAME columns and that you are *not* entering values in any other columns; the other columns will contain null values.

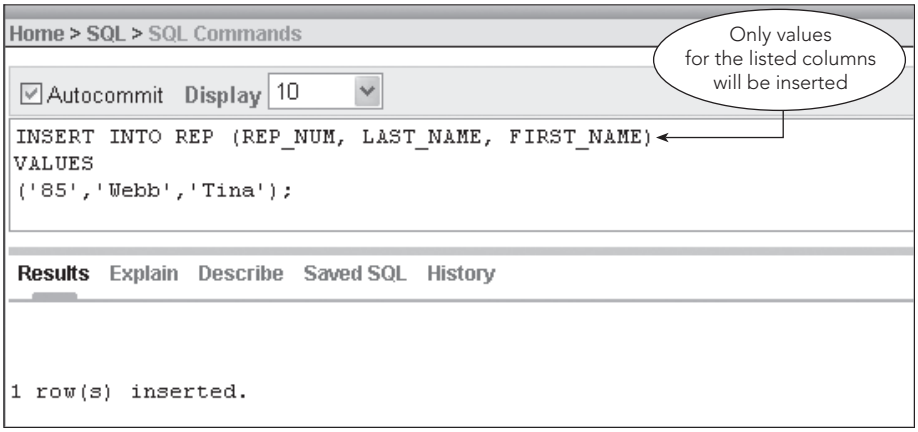

FIGURE 3-15 Inserting a row in the REP table that contains null values

VIEWING TABLE DATA

To view the data in a table, you use the **SELECT** command, which is described in more detail in Chapters 4 and 5.

EXAMPLE 5

Display all the rows and columns in the REP table.

You can use a simple version of the SELECT command to display all the rows and columns in a table by typing the word SELECT, followed by an asterisk, followed by the word FROM and the name of the table containing the data you want to view. Just as with other SQL commands, the command ends with a semicolon. In Oracle, you type the command shown in Figure 3-16, and then click the Run button to display the results.

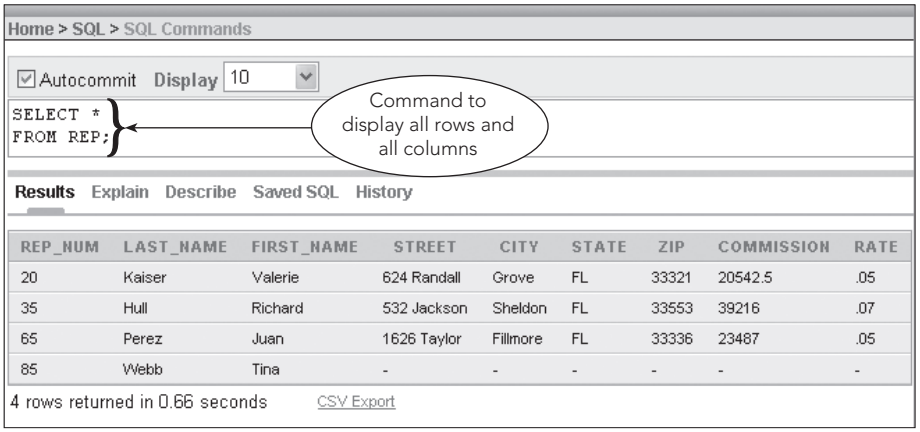

FIGURE 3-16 Using a SELECT command to view table data

ACCESS USER NOTE

In Access, type the query shown in Figure 3-17 in SQL view to display all the rows and columns in a table.

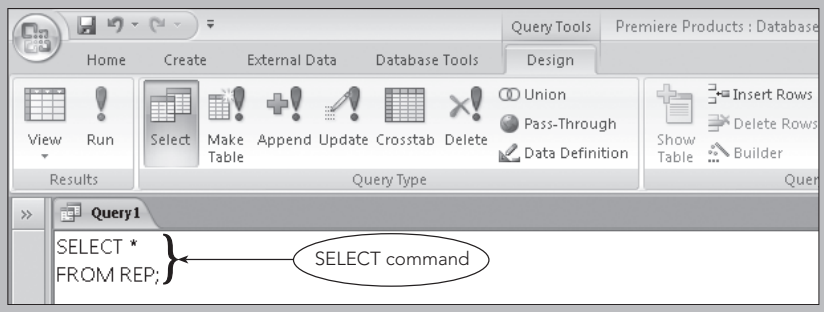

FIGURE 3-17 Using a SELECT command to view table data in Access

To run the query, click the Run button. Access will display the query results in Datasheet view, as shown in Figure 3-18. If the data does not fit on the screen, you can adjust the columns to best fit the data they contain by double-clicking the right edge of each column heading. You can use the scroll bars when necessary to view data that has scrolled off the screen.

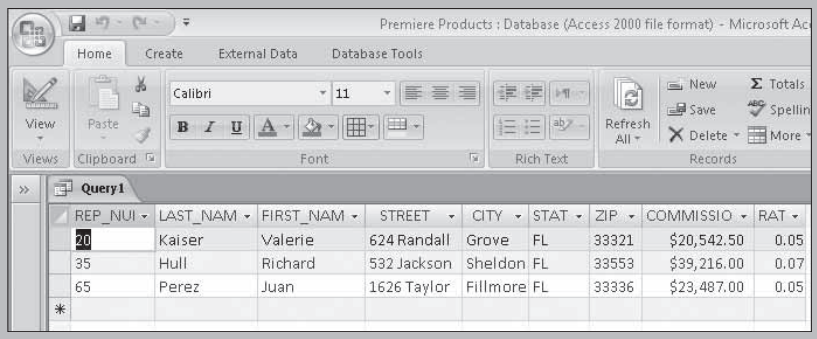

FIGURE 3-18 Query results in Access Datasheet view

SQL SERVER USER NOTE

In SQL Server, type the query shown in Figure 3-19 in the Query Editor window.

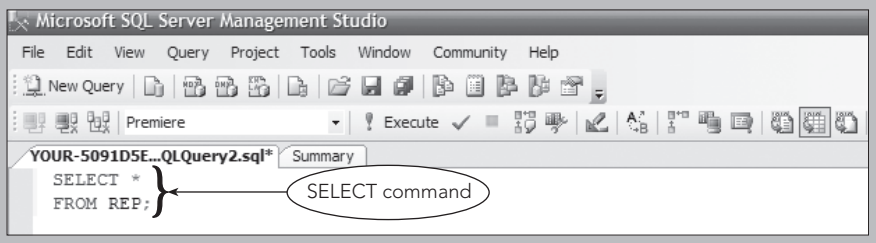

FIGURE 3-19 Using a SELECT command to view table data in SQL Server

To run the query, click the Execute button. SQL Server will display the query results in the Results pane, as shown in Figure 3-20. If the data does not fit on the screen, you can adjust the columns to best fit the data they contain by dragging the right edge of each column heading to make it narrower or wider.

Results | Messages

	REP_NUM	LAST_NAME	FIRST_NAME	STREET	CITY	STA...	ZIP	COMMISSION	RATE
1	20	Kaiser	Valerie	624 Randall	Grove	FL	33321	20542.50	0.05
2	35	Hull	Richard	532 Jackson	Sheldon	FL	33553	39216.00	0.07
3	65	Perez	Juan	1626 Taylor	Fillmore	FL	33336	23487.00	0.05

FIGURE 3-20 Query results in SQL Server

CORRECTING ERRORS IN A TABLE

After executing a SELECT command to view a table's data, you might find that you need to change the value in a column. You can use the **UPDATE** command shown in Figure 3-21 to change a value in a table. The UPDATE command shown in Figure 3-21 changes the last name in the row on which the sales rep number is 85 to Perry.

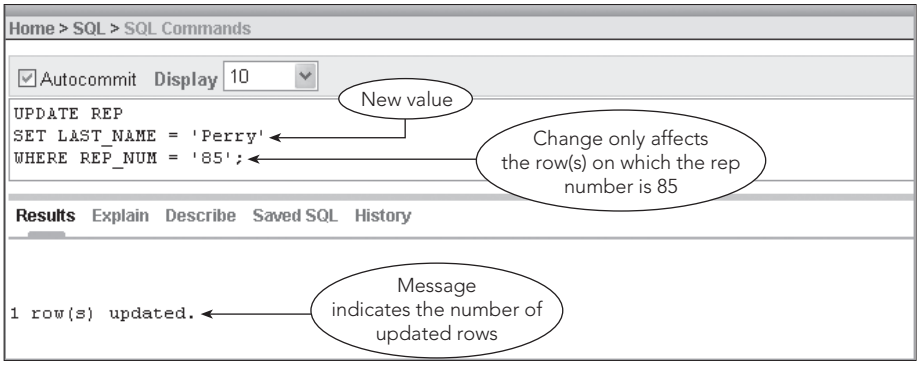

FIGURE 3-21 Using an UPDATE command to change a value

The SELECT command shown in Figure 3-22 displays the results of the UPDATE command shown in Figure 3-21, in which the last name for rep number 85 is Perry.

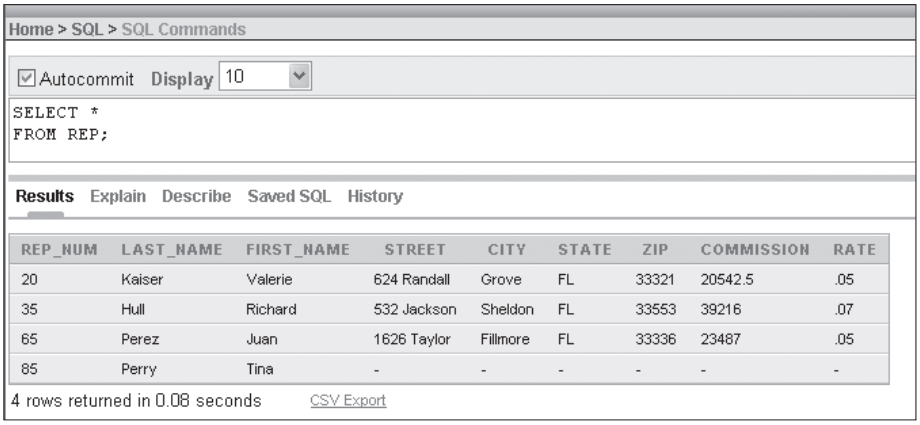

FIGURE 3-22 Last name changed for sales rep number 85

When you need to delete a row from a table, you can use the **DELETE** command. The DELETE command shown in Figure 3-23 deletes any row on which the sales rep number is 85.

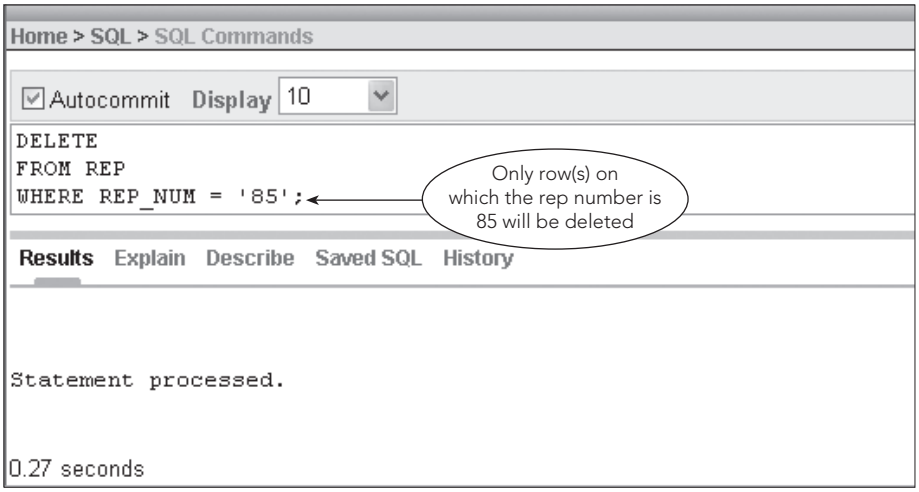

FIGURE 3-23 Using a DELETE command to delete a row

The SELECT command shown in Figure 3-24 displays the updated data.

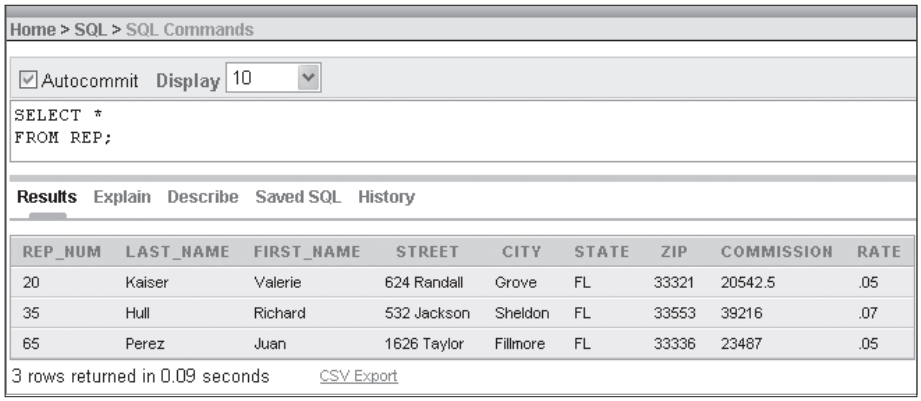

FIGURE 3-24 Sales rep number 85 deleted from REP table

Q & A

Question: How do I correct errors in my data?
Answer: The method you use to correct an error depends on the type of error you need to correct. If you added a row that should not be in the table, use a DELETE command to remove it. If you forgot to add a row, you can use an INSERT command to add it. If you added a row that contains incorrect data, you can use an UPDATE command to make the necessary corrections. Alternatively, you could use a DELETE command to remove the row containing the error and then use an INSERT command to insert the correct row.

SAVING SQL COMMANDS

Oracle lets you save a command so you can use it again without retyping it. In many DBMSs, you save commands in a **script file**, or simply a **script**, which is a text file with the .sql filename extension. When you use Oracle to create a script, Oracle stores the script in a special location called the **script repository**. If you want to save a script on the local file system, such as on a hard drive or USB drive, you can do so by downloading the script. When you need to use a script that is stored on the local file system, but is not currently stored in Oracle, you can upload the script so you can use it in Oracle. The following steps describe how to create and use scripts in the Oracle Database Express Edition. If you are using a different version of Oracle or another DBMS, use Help or consult the system documentation to determine how to accomplish the same tasks.

To create a script:

1. Load the Oracle Database Express Edition home page and log in.
2. Click the SQL icon arrow, point to SQL Scripts, and then click Create. The Script Editor page opens.
3. In the Script Name text box, type a name for the script.

4. Click in the text box on the page to activate it, and then type the command or commands to save in the script. When necessary, click the Run button to execute the commands saved in the script.
5. When you are finished, click the Save button. You return to the SQL Scripts page and the script you created appears as an icon on the page.

To view, edit, or run an existing script:

1. Load the Oracle Database Express Edition home page and log in.
2. Click the SQL icon arrow, point to SQL Scripts, and then click View.
3. Click the icon for the desired script. The script appears on the Script Editor page. You can use this page to view the content of the script or to make changes to it by editing the commands it contains. If you edit a script, click the Save button to save your changes.
4. To run a script, click the Run button. The Run Script page loads and asks you to confirm running the script. Click the Run button on the Run Script page. The Manage Script Results page opens and displays the script name and an icon in the View Results column. To see the results of the command stored in the script, click the icon in the View Results column.

When you are finished using a script or no longer need to store it, you can delete it. To delete a script:

1. Follow the previous instructions to view the script.
2. Click the Delete button on the Script Editor page.
3. Click the OK button to confirm the deletion.

To download a script from the script repository so you can save it as a file:

1. Follow the previous instructions to view the script.
2. Click the Download button. The File Download dialog box opens.
3. In the dialog box, click the Save button, and then save the file to the desired location.
4. Click the Close button to close the Download complete dialog box.

To upload a script to the script repository:

1. Load the Oracle Database Express Edition home page and log in.
2. Click the SQL icon arrow, point to SQL Scripts, and then click Upload. The Upload Script page opens.
3. Click the Browse button. The Choose file dialog box opens. Navigate to and select the script file to upload. Click the Open button. (If you want to upload the script with a different filename, type the new name in the Script Name text box.)
4. On the Upload Script page, click the Upload button. An icon for the script appears on the SQL Scripts page.

ACCESS USER NOTE

Access does not use script files, but you can save an SQL command as a query object in the database. Open the database, create the query in SQL view, click the Save button on the Quick Access toolbar (or on the menu bar), and then save the query with the desired object name. To run the query without first viewing the SQL command, right-click the query in the Navigation Pane (or in the Database window), and then click Open on the shortcut menu. The query results will appear in Datasheet view. If you want to view the SQL command before running the query, right-click the query in the Navigation Pane, and then click Design View on the shortcut menu. To run the SQL command, click the Run button.

SQL SERVER USER NOTE

SQL Server can store scripts in any folder on your local system. All scripts created in SQL Server are text files with the .sql extension. To create a script file in SQL Server:

1. Load SQL Server Management Studio, and then click the Connect button in the Connect to Server dialog box.
2. Open the appropriate database, and then click the New Query button.
3. Type the command or commands to save in the script. When necessary, click the Execute button to execute the commands saved in the script.
4. When you are finished, click the Save button, navigate to the location in which to save the script, and then enter a name for the script.

To view, edit, or run an existing script:

1. Load SQL Server Management Studio, and then click the Connect button in the Connect to Server dialog box.
2. Open the appropriate database, and then click the New Query button.
3. Click the Open File button on the toolbar.
4. Navigate to the folder containing the script file, select the script file, and then click the Open button in the Open File dialog box. (If the Connect to Database Engine dialog box opens, click the Connect button.) The script appears in the Query Editor window. You can view the contents of the script to make changes to it by editing the commands. If you edit a script, click the Save button to save your changes.
5. To run a script, click the Execute button.

CREATING THE REMAINING DATABASE TABLES

To create the remaining tables in the Premiere Products database, you need to execute the appropriate CREATE TABLE and INSERT commands. You should save these commands as scripts so you can re-create your database, if necessary, by running the scripts.

NOTE

Your instructor might give you the script files to use to create the tables in the Premiere Products, Henry Books, and Alexamara Marina Group databases and to insert data into them.

Figure 3-25 shows the CREATE TABLE command for the CUSTOMER table. Notice that the CUSTOMER_NAME column is specified as NOT NULL. Additionally, the CUSTOMER_NUM column is the table's primary key, indicating that the CUSTOMER_NUM column is the unique identifier of rows in the table. With this column designated as the primary key, the DBMS will reject any attempt to store a customer number that already exists in the table.

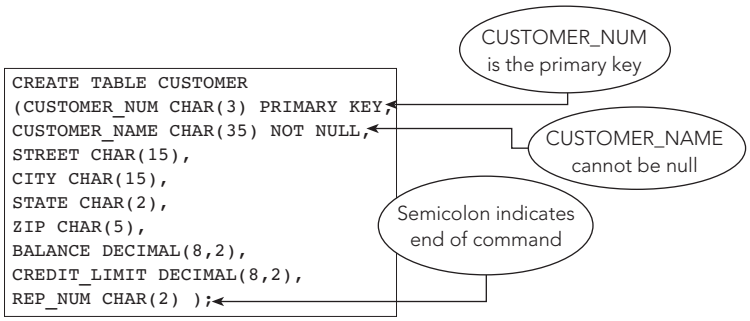

FIGURE 3-25 CREATE TABLE command for the CUSTOMER table

After creating the CUSTOMER table, you can create another script file containing the INSERT commands to add the customer rows to the table. When a script file contains more than one command, each command must end with a semicolon. Figure 3-26 shows the INSERT commands to add rows to the CUSTOMER table. As noted previously, to enter an apostrophe (single quotation mark) in the value for a field, type two single quotation marks, as illustrated in the name in the first INSERT command (Al's Appliance and Sport) in Figure 3-26.

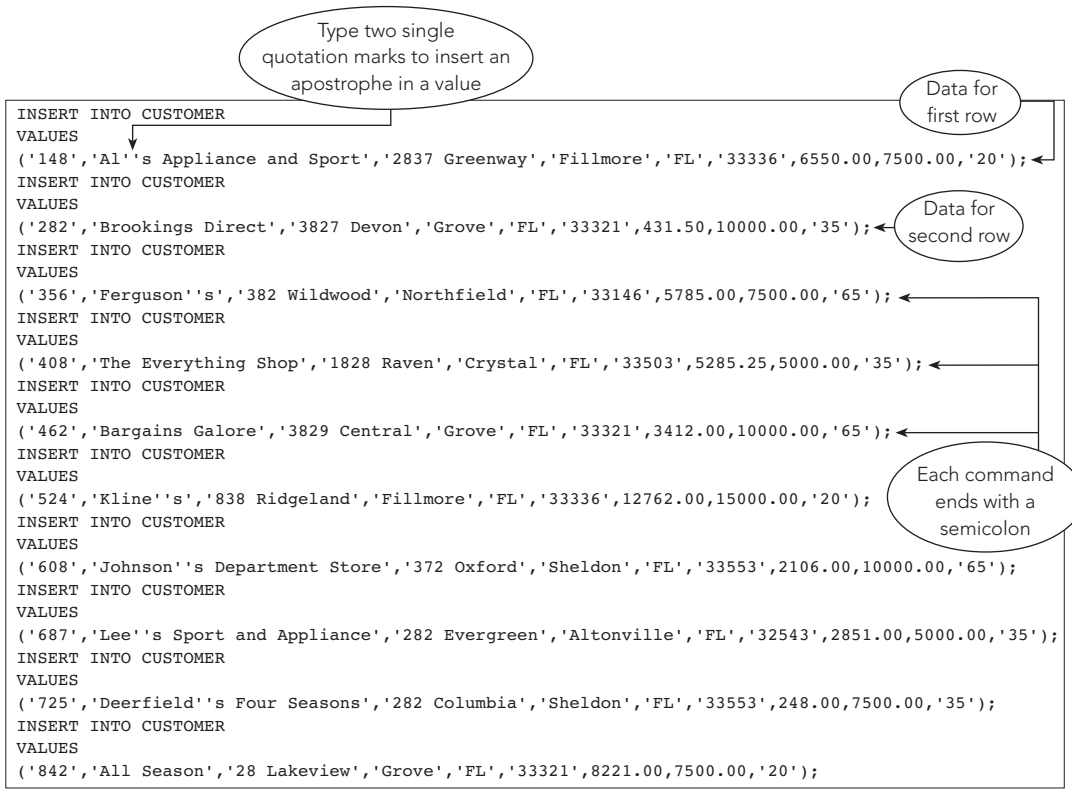

FIGURE 3-26 INSERT commands for the CUSTOMER table

Figures 3-27 through 3-32 show the scripts for the CREATE TABLE and INSERT commands for creating and inserting data into the ORDERS, PART, and ORDER_LINE tables in the Premiere Products database. Figure 3-27 contains the CREATE TABLE command for the ORDERS table.

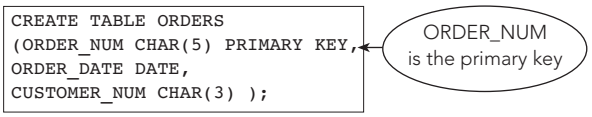

FIGURE 3-27 CREATE TABLE command for the ORDERS table

Figure 3-28 contains the INSERT commands to load data into the ORDERS table. Notice the way that dates are entered.

```
INSERT INTO ORDERS
VALUES
('21608','20-OCT-2010','148');
INSERT INTO ORDERS
VALUES
('21610','20-OCT-2010','356');
INSERT INTO ORDERS
VALUES
('21613','21-OCT-2010','408');
INSERT INTO ORDERS
VALUES
('21614','21-OCT-2010','282');
INSERT INTO ORDERS
VALUES
('21617','23-OCT-2010','608');
INSERT INTO ORDERS
VALUES
('21619','23-OCT-2010','148');
INSERT INTO ORDERS
VALUES
('21623','23-OCT-2010','608');
```

Format used to enter a date in Oracle

FIGURE 3-28 INSERT commands for the ORDERS table

Figure 3-29 contains the CREATE TABLE command for the PART table.

```
CREATE TABLE PART
(PART_NUM CHAR(4) PRIMARY KEY,
DESCRIPTION CHAR(15),
ON_HAND DECIMAL(4,0),
CLASS CHAR(2),
WAREHOUSE CHAR(1),
PRICE DECIMAL(6,2) );
```

PART_NUM is the primary key

FIGURE 3-29 CREATE TABLE command for the PART table

Figure 3-30 contains the INSERT commands to load data into the PART table.

```
INSERT INTO PART
VALUES
('AT94','Iron',50,'HW','3',24.95);
INSERT INTO PART
VALUES
('BV06','Home Gym',45,'SG','2',794.95);
INSERT INTO PART
VALUES
('CD52','Microwave Oven',32,'AP','1',165.00);
INSERT INTO PART
VALUES
('DL71','Cordless Drill',21,'HW','3',129.95);
INSERT INTO PART
VALUES
('DR93','Gas Range',8,'AP','2',495.00);
INSERT INTO PART
VALUES
('DW11','Washer',12,'AP','3',399.99);
INSERT INTO PART
VALUES
('FD21','Stand Mixer',22,'HW','3',159.95);
INSERT INTO PART
VALUES
('KL62','Dryer',12,'AP','1',349.95);
INSERT INTO PART
VALUES
('KT03','Dishwasher',8,'AP','3',595.00);
INSERT INTO PART
VALUES
('KV29','Treadmill',9,'SG','2',1390.00);
```

FIGURE 3-30 INSERT commands for the PART table

Figure 3-31 contains the CREATE TABLE command for the ORDER_LINE table. Notice the way that the primary key is defined when it consists of more than one column.

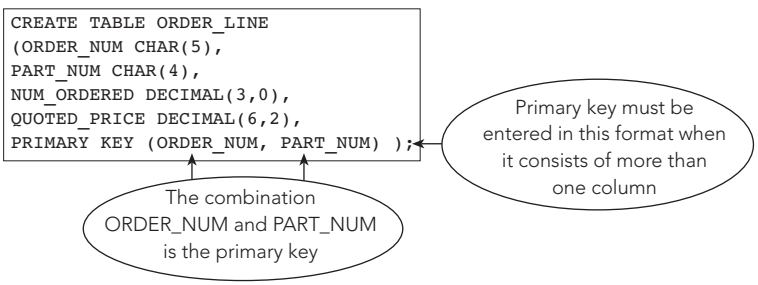

FIGURE 3-31 CREATE TABLE command for the ORDER_LINE table

Figure 3-32 contains the INSERT commands to load data into the ORDER_LINE table.

```
INSERT INTO ORDER_LINE
VALUES
('21608','AT94',11,21.95);
INSERT INTO ORDER_LINE
VALUES
('21610','DR93',1,495.00);
INSERT INTO ORDER_LINE
VALUES
('21610','DW11',1,399.99);
INSERT INTO ORDER_LINE
VALUES
('21613','KL62',4,329.95);
INSERT INTO ORDER_LINE
VALUES
('21614','KT03',2,595.00);
INSERT INTO ORDER_LINE
VALUES
('21617','BV06',2,794.95);
INSERT INTO ORDER_LINE
VALUES
('21617','CD52',4,150.00);
INSERT INTO ORDER_LINE
VALUES
('21619','DR93',1,495.00);
INSERT INTO ORDER_LINE
VALUES
('21623','KV29',2,1290.00);
```

FIGURE 3-32 INSERT commands for the ORDER_LINE table

DESCRIBING A TABLE

The CREATE TABLE command defines a table's structure by listing its columns, data types, and column lengths. The CREATE TABLE command also indicates which columns cannot accept nulls. When you work with a table, you might not have access to the CREATE TABLE command that was used to create it. For example, another programmer might have created the table, or perhaps you created the table several months ago but did not save the command. You might want to examine the table's structure to see the details about the columns in the table. Each DBMS provides a method to examine a table's structure.

EXAMPLE 6

Describe the REP table.

In Oracle, you can use the **DESCRIBE** command to list all the columns in a table and their properties. Figure 3-33 shows the DESCRIBE command for the REP table. The result indicates the name of each column in the table, along with its data type and length. A value of 1 in the Primary Key column indicates the table's primary key column. A check mark in the Nullable column indicates a column that can accept null values. (The Precision, Scale, Default, and Comment columns in the results are beyond the scope of this discussion.)

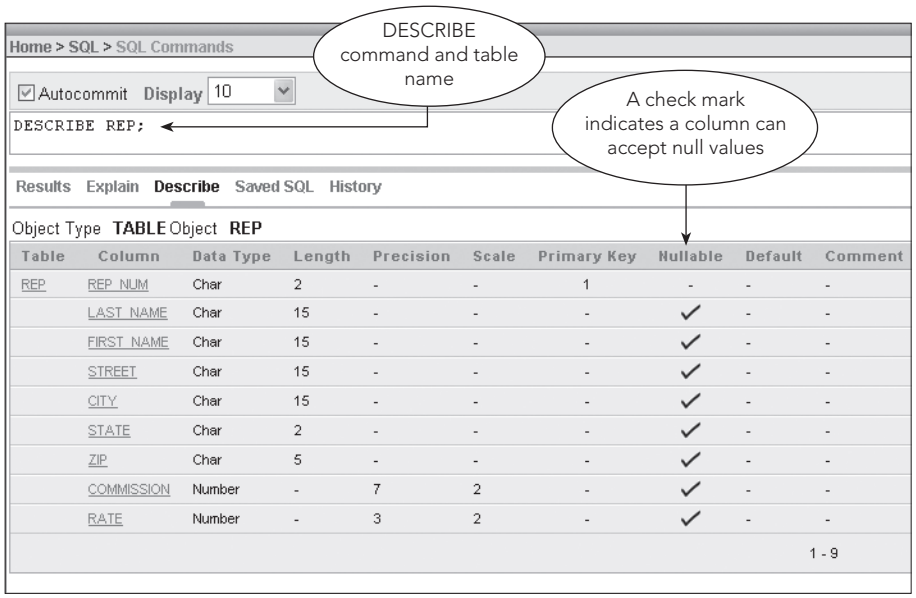

FIGURE 3-33 DESCRIBE command for the REP table

ACCESS USER NOTE

In Access, you use the Documenter tool to describe the tables (and other objects) in a database. To start the Documenter in Access 2003, open the database, click Tools on the menu bar, point to Analyze, and then click Documenter. To start the Documenter in Access 2007, click the Database Tools tab on the Ribbon, and then click the Database Documenter button in the Analyze group. In the Documenter dialog box, click the Tables tab, select the tables that you want to describe by putting a check mark in the check box next to their names, and then click the OK button. The Object Definition window opens and displays a report containing the requested documentation. You can customize the Documenter to control the amount of detail included in the report.

SQL SERVER USER NOTE

In SQL Server, you execute the sp_columns command to list all the columns in a table. The following command will list all the columns in the REP table:

```
Exec sp_columns REP
```

The result will indicate the name of each column in the REP table, along with its data type and length. A value of 1 in the Nullable column indicates a column that can accept null values. (The remaining columns that appear in the results are beyond the scope of this discussion.)

Chapter Summary

- Use the CREATE TABLE command to create a table by typing the table name and then listing within a single set of parentheses the columns in the table.
- Use the DROP TABLE command to delete a table and all its data from the database.
- Some commonly used data types in are INT, SMALLINT, DECIMAL, CHAR, VARCHAR, and DATE. Microsoft Access does not support DECIMAL. SQL Server uses DATETIME instead of DATE.
- A null data value (or null) is a special value that is used when the actual value for a column is unknown, unavailable, or not applicable.
- Use the NOT NULL clause in a CREATE TABLE command to identify columns that cannot accept null values.
- Use the INSERT command to insert rows into a table.
- Use the SELECT command to view the data in a table.
- Use the UPDATE command to change the value in a column.
- Use the DELETE command to delete a row from a table.
- You can save SQL commands in a script file in Oracle and SQL Server. In Microsoft Access, you save the commands as a query object in the database.
- You can use the DESCRIBE command in Oracle to display a table's structure and layout. In Access, you use the Documenter tool to produce a report of a table's structure and layout. In SQL Server, execute the sp_columns command to display the structure and layout of a table.

Key Terms

breadcrumb
CREATE TABLE
data type
DELETE
DESCRIBE
DROP TABLE
INSERT
NOT NULL
null
null data value
script
script file
script repository
SELECT
Structured Query Language (SQL)
UPDATE

Review Questions

1. How do you create a table using SQL?
2. How do you delete a table using SQL?
3. What are the common data types used to define columns using SQL?

4. Identify the best data type to use to store the following data in Oracle, in SQL Server, and in Access:
   a. The month, day, and year that an employee was hired
   b. An employee's Social Security number
   c. The department in which an employee works
   d. An employee's hourly pay rate
5. Write a paragraph that explains the difference between the CHAR data type and the VARCHAR data type. Use your Web browser and your favorite search engine to find examples of when to use VARCHAR and when to use CHAR. Be sure to cite the URL(s) that provided the examples as references at the end of your document.
6. What is a null value? How do you use SQL to identify columns that cannot accept null values?
7. Which SQL command do you use to add a row to a table?
8. Which SQL command do you use to view the data in a table?
9. Which SQL command do you use to change the value in a column in a table?
10. Which SQL command do you use to delete rows from a table?
11. How do you display the columns in a table and their characteristics in Oracle?

Exercises

To print a copy of your commands and results using Oracle, use the browser's Print command on the File menu or click the Print button on the browser's toolbar.

To print a copy of your commands and results using Access, use the instructions provided in the chapter to save your commands as query objects. To print a command, start Word or another word processor and create a new document. Select the SQL command in Access, copy it to the Clipboard, and then paste it into the document. To copy and paste a command's results, right-click the datasheet selector (the box in the upper-left corner of the datasheet) to select the entire datasheet, copy it to the Clipboard, and then paste it into the document.

To print a copy of your commands and results using SQL Server, start Word or another word processor and create a new document. Select the SQL command in SQL Server, copy it to the Clipboard, and then paste it into the document. To copy and paste a command's results, right-click the datasheet selector (the box in the upper-left corner of the datasheet) to select the entire datasheet, copy it to the Clipboard, and then paste it into the document.

Premiere Products

Use SQL to complete the following exercises.

1. Create a table named SALES_REP. The table has the same structure as the REP table shown in Figure 3-6 except the LAST_NAME column should use the VARCHAR data type and the COMMISSION and RATE columns should use the NUMBER data type. Execute the command to describe the layout and characteristics of the SALES_REP table.

2. Add the following row to the SALES_REP table: rep number: 25, last name: Lim; first name: Louis; street: 535 Vincent; city: Grove; state: FL; zip code: 33321; commission: 0.00; and rate: 0.05. Display the contents of the SALES_REP table.
3. Delete the SALES_REP table.
4. Run the script file for the Premiere database to create the five tables and add records to the tables. Be sure to select the file for the particular DBMS that you are using (Oracle, SQL Server, or Access). (*Note:* If you do not have the files for this text, ask your instructor for assistance.)
5. Confirm that you have created the tables correctly by describing each table and comparing the results to Figures 3-6, 3-25, 3-27, 3-29, and 3-31.
6. Confirm that you have added all data correctly by viewing the data in each table and comparing the results to Figure 2-1 in Chapter 2.

Henry Books

Use SQL to complete the following exercises.

1. Create a table named SALES_BRANCH. The table has the same structure as the BRANCH table shown in Figure 3-34 except the BRANCH_LOCATION column should use the VARCHAR data type and the BRANCH_NUM and NUM_EMPLOYEES columns should use the NUMBER data type. Execute the command to describe the layout and characteristics of the SALES_BRANCH table.
2. Add the following row to the SALES_BRANCH table: branch number: 5; branch name: Henry Town Plaza; branch location: 165 Plaza; and number of employees: 3. Display the contents of the SALES_BRANCH table.
3. Delete the SALES_BRANCH table.
4. Run the script file for the Henry Books database to create the six tables and add records to the tables. Be sure to select the file for the particular DBMS that you are using (Oracle, SQL Server, or Access). (*Note:* If you do not have the files for this text, ask your instructor for assistance.)
5. Confirm that you have created the tables correctly by describing each table and comparing the results to Figure 3-34.
6. Confirm that you have added all data correctly by viewing the data in each table and comparing the results to Figures 1-4 through 1-7 in Chapter 1.

BRANCH

Column	Type	Length	Decimal places	Nulls allowed?	Description
BRANCH_NUM	DECIMAL	2	0	No	Branch number (primary key)
BRANCH_NAME	CHAR	50			Branch name
BRANCH_LOCATION	CHAR	50			Branch location
NUM_EMPLOYEES	DECIMAL	2	0		Number of employees

PUBLISHER

Column	Type	Length	Decimal places	Nulls allowed?	Description
PUBLISHER_CODE	CHAR	3		No	Publisher code (primary key)
PUBLISHER_NAME	CHAR	25			Publisher name
CITY	CHAR	20			Publisher city

AUTHOR

Column	Type	Length	Decimal places	Nulls allowed?	Description
AUTHOR_NUM	DECIMAL	2	0	No	Author number (primary key)
AUTHOR_LAST	CHAR	12			Author last name
AUTHOR_FIRST	CHAR	10			Author first name

BOOK

Column	Type	Length	Decimal places	Nulls allowed?	Description
BOOK_CODE	CHAR	4		No	Book code (primary key)
TITLE	CHAR	40			Book title
PUBLISHER_CODE	CHAR	3			Publisher code
TYPE	CHAR	3			Book type
PRICE	DECIMAL	4	2		Book price
PAPERBACK	CHAR	1			Paperback (Y, N)

FIGURE 3-34 Table layouts for the Henry Books database

WROTE

Column	Type	Length	Decimal places	Nulls allowed?	Description
BOOK_CODE	CHAR	4		No	Book code (primary key)
AUTHOR_NUM	DECIMAL	2	0	No	Author number (primary key)
SEQUENCE	DECIMAL	1	0		Sequence number

INVENTORY

Column	Type	Length	Decimal places	Nulls allowed?	Description
BOOK_CODE	CHAR	4		No	Book code (primary key)
BRANCH_NUM	DECIMAL	2	0	No	Branch number (primary key)
ON_HAND	DECIMAL	2	0		Units on hand

FIGURE 3-34 Table layouts for the Henry Books database (continued)

Alexamara Marina Group

Use SQL to complete the following exercises.

1. Create a table named BOAT_SLIP. The table has the same structure as the MARINA_SLIP table shown in Figure 3-35 except the SLIP_ID, LENGTH, and RENTAL_FEE columns should use the NUMBER data type. Execute the command to describe the layout and characteristics of the BOAT_SLIP table.
2. Add the following record to the BOAT_SLIP table: slip ID: 12; marina number: 2; slip number: 7; length: 25; rental fee: 1800; boat name: Bavant; boat type: Ray 25; and owner number: FL13. Display the contents of the BOAT_SLIP table.
3. Delete the BOAT_SLIP table.
4. Run the script file for the Alexamara Marina Group database to create the five tables and add records to the tables. Be sure to select the file for the particular DBMS that you are using (Oracle, SQL Server, or Access). (*Note:* If you do not have the files for this text, ask your instructor for assistance.)
5. Confirm that you have created the tables correctly by describing each table and comparing the results to Figure 3-35.
6. Confirm that you have added all data correctly by viewing the data in each table and comparing the results to Figures 1-8 through 1-12 in Chapter 1.

MARINA

Column	Type	Length	Decimal places	Nulls allowed?	Description
MARINA_NUM	CHAR	4		No	Marina number (primary key)
NAME	CHAR	20			Marina name
ADDRESS	CHAR	15			Marina street address
CITY	CHAR	15			Marina city
STATE	CHAR	2			Marina state
ZIP	CHAR	5			Marina zip code

OWNER

Column	Type	Length	Decimal places	Nulls allowed?	Description
OWNER_NUM	CHAR	4		No	Owner number (primary key)
LAST_NAME	CHAR	50			Owner last name
FIRST_NAME	CHAR	20			Owner first name
ADDRESS	CHAR	15			Owner street address
CITY	CHAR	15			Owner city
STATE	CHAR	2			Owner state
ZIP	CHAR	5			Owner zip code

MARINA_SLIP

Column	Type	Length	Decimal places	Nulls allowed?	Description
SLIP_ID	DECIMAL	4	0	No	Slip ID (primary key)
MARINA_NUM	CHAR	4			Marina number
SLIP_NUM	CHAR	4			Slip number in the marina
LENGTH	DECIMAL	4	0		Length of slip (in feet)
RENTAL_FEE	DECIMAL	8	2		Annual rental fee for the slip
BOAT_NAME	CHAR	50			Name of boat currently in the slip
BOAT_TYPE	CHAR	50			Type of boat currently in the slip
OWNER_NUM	CHAR	4			Number of boat owner renting the slip

FIGURE 3-35 Table layouts for the Alexamara Marina Group database

SERVICE_CATEGORY

Column	Type	Length	Decimal places	Nulls allowed?	Description
CATEGORY_NUM	DECIMAL	4	0	No	Category number (primary key)
CATEGORY_ DESCRIPTION	CHAR	255			Category description

SERVICE_REQUEST

Column	Type	Length	Decimal places	Nulls allowed?	Description
SERVICE_ID	DECIMAL	4	0	No	Service ID (primary key)
SLIP_ID	DECIMAL	4	0		Slip ID of the boat for which service is requested
CATEGORY_NUM	DECIMAL	4	0		Category number of the requested service
DESCRIPTION	CHAR	255			Description of specific service requested for boat
STATUS	CHAR	255			Description of status of service request
EST_HOURS	DECIMAL	4	2		Estimated number of hours required to complete the service
SPENT_HOURS	DECIMAL	4	2		Hours already spent on the service
NEXT_SERVICE_DATE	DATE				Next scheduled date for work on this service (or null if no next service date is specified)

FIGURE 3-35 Table layouts for the Alexamara Marina Group database (continued)

1. Use the CREATE TABLE command to create a table by typing the table name and then listing within a single set of parentheses the columns in the table.
3. CHAR, VARCHAR, DATE, DECIMAL, INT, SMALLINT
5. Answers will vary. Answers should mention that the difference between CHAR and VARCHAR is that CHAR is fixed length, while VARCHAR is variable length. This means that CHAR is always the same size and takes up the same amount of bytes, while VARCHAR varies. VARCHAR is a good choice when you are storing email addresses and comments that can vary in size.
7. Use the INSERT command.
9. Use the UPDATE command.
11. Use the DESCRIBE command.

This page contains answers for this chapter only.